Um caso de estudo do evento Direct Path Reads. Por que ele é tão importante para o Oracle Exadata?

Por Joel Perez & Flávio Soares (OCE)
Postado em Janeiro 2015

Revisado por Marcelo Pivovar - Solution Architect

Você DBA provavelmente já deve ter ouvido falar do Oracle Smart Scan. Pois bem, o Smart Scan realmente é um dos segredos por trás da extrema velocidade de processamento das instruções dentro do Oracle Exadata Machine. Existe também o conceito de offloading dos dados dentro da arquitetura do Exadata, que refere ao fato do processamento dos dados ser feito a nível da camada de storage (storage layer) e não mais dentro da camada de banco de dados (database layer). O Smart Scan por outro lado é mais focado a nível de SQL e não de dados como o Offloading, mais como podemos ver nos documentos e manuais, a Oracle acabou que juntando esses dois conceitos e chamando apenas de “Smart Scan”.

Apesar de toda velocidade de processamento que a máquina Exadata possuí, ela não seria o que é sem a capacidade de realização de Offloading e Smart Scan. Sem essas features, o Exadata seria apenas mais uma máquina de alto poder de processamento, porém sem grande inteligência de manipulação dos dados … e como vemos, é essa inteligência que faz toda a diferença.

Mais afinal, o que o “Direct Path Reads” tem haver com o Offloading de Dados e o Exadata Smart Scan? A resposta para essa pergunta é simples: O offloading e/ou Smart Scan não acontecerá, caso sua instrução não utilize o Direct Path Reads.

Dois pré-requisitos básicos, são necessários para o Offloading/Smart Scan:

1. Obviamente, realizar a instrução em um storage Exadata.
2. Realizar o “Direct Path Reads” na sua instrução.

É claro que isso envolve algumas restrições, mais basicamente tendo esses requisitos, a sua consulta irá acabar realizando o Smart Scan.

O que é Direct Path Reads?

O Direct Path Reads foi criado pela Oracle para ignorar o Buffer Cache. O Buffer Cache como você já deve saber, é uma área da SGA destinada a manter os blocos recentes lidos, assim todos os usuários conectados na instância é capaz de ler e compartilhar desse cache sem a necessidade de ler esses blocos novamente do disco. Isso é um excelente ganho de performance, já que o evitamos o acesso a disco sempre que uma instrução é realizada. A Oracle fez um excelente trabalho ao longo dos anos, aperfeiçoando cada vez mais esse cache através dos algoritmos LRU e MRU, veja mais aqui: http://docs.oracle.com/cd/B28359_01/server.111/b28318/memory.htm#CNCPT1224

Realmente existe muito mais vantagem do que desvantagem em utilizar o Buffer Cache, porém a grande desvantagem, por ser um processo automático de gerenciamento de Buffer, o Oracle acaba por colocar “sujeiras” dentro desse cache, removendo dados que inclusive eram mais acessados pelas demais sessões. Imagina esse caso por exemplo, um relatório que é disparado uma única vez no mês para cálculos de fechamento que movimenta uma enorme quantidade de dados, por qual razão você gostaria de colocar todos esses dados gerados do relatório dentro do buffer cache do seu banco, sendo que essa instrução será executada apenas uma única vez e não será compartilhado com outras sessões dentro do banco. Inclusive, todo esses dados gerados, pode ser maior do que o próprio buffer cache, causando assim um extremo overhead em remover dados mais acessados e adicionar dados que nunca irá ser acessado. Será um tremendo trabalho em alocar, desalocar e realocar tudo novamente.

Foi aí que surgiu o Direct Path Reads.

O mecanismo de Direct Path Reads já está disponível no kernel do Oracle há muito tempo. Ele foi inicialmente implementado para trabalhar exclusivamente com os processos slaves sempre que uma instrução era disparada via paralelismo. Como os processos paralelos, como via de regra, devem ler grandes quantidades de dados o Direct Path Reads entrou na jogada para ignora completamente o mecanismo padrão do buffer. Foi decidido a partir daí, que os blocos deveriam ser armazenados em suas próprias memórias (PGA) e não mais na SGA quando se utiliza-se a consulta via DPR.

De acordo com o metalink, a partir do Oracle 11gR2, o kernel foi modificado para decidir realizar mais Direct Path Reads do que na versão 10g, ou seja na versão 10g o serial table scans tem muito mais chance de ser realizado no buffer compartilhado (scattered reads) do que na própria memória do processo (direct path reads).

Como identifico o evento Direct Path Reads?

Existem várias formas de se identificar o evento “Direct Path Reads”, uma delas é através das views de wait’s do Oracle, como por exemplo a *v$session_wait*.

A view *v$session_wait* mostra sempre o atual evento de espera ocorrido pela sessão. Usando o SQL abaixo, podemos identificar através da coluna EVENT, a utilização do evento “direct path read” para a consulta em questão (sql_id).

```
SELECT  s.sid, w.state, w.event, s.sql_id, s.sql_child_number,
        w.seq#, w.seconds_in_wait, w.p1text||'=  '||w.p1 p1,
        w.p2text||'= '||w.p2 p2, w.p3text||'=  '||w.p3 p3
FROM  v$session s, v$session_wait w
WHERE  w.sid = s.sid AND w.sid = ""
```

Vamos a uma prova de teste. A partir de agora, vou utilizar incessantemente o parâmetro oculto chamado “_serial_direct_read” para forçar a utilização do direct reads. Vou falar mais desse parâmetro mais a frente, o importante agora é saber que através dele podemos forçar a utilização do evento Direct Path Reads.

Os testes serão feito através da tabela fss.hsk1. Essa tabela, nada mais é do que uma tabela de teste que sempre utilizo em meus testes com Oracle. A tabela é criada dentro do owner FSS e contém cerca de 4G (você pode mudar o tamanho da tabela criada, alterando a quantidade de linhas inseridas na tabela, veja os comentários dentro do script). Através dos links abaixo, você poderá utilizar também a mesma tabela que vou demonstrar os testes a seguir.

Criação do usuário FSS: fss_user.sql
Criação das tabelas do usuário FSS: fss_create_tables.sql
Lembrando também, que todos esses testes foram feitos em um Oracle 11.2.0.3.

Identificando o uso do Direct Read, através da view de espera *v$session_wait*

Vamos ao teste que interessa. Com a sessão 1, iremos executar a seguinte instrução SQL na tabela fss.hsk1. Primeiro vamos definir o parâmetro oculto _serial_direct_read para ALWAYS, dessa forma eu estou forçando com que todas as minhas consultas sejam executadas via “Direct Path Reads”

```
SQL>  ALTER SESSION SET "_serial_direct_read"=ALWAYS;
Session  altered.
SQL>  select avg(length(col1) + length(col2)) from fss.hsk1 where col3 > 1;
```

Rapidamente, enquanto executa a consulta acima, com a sessão 2, vamos ver através da v$session_wait que o evento de espera atual é o “direct path”:

```
-->  SESSÃO 2
```

```
SQL>  SELECT
      1  s.sid, w.state, w.event, s.sql_id,  s.sql_child_number, w.seq#
      2  FROM v$session s, v$session_wait w
      3  WHERE w.sid = s.sid AND w.sid=152;


SID       STATE               EVENT                         SQL_ID         CH#  SEQ#
--------  ------------------- ----------------------------- ------------- ----- -------
152       WAITED SHORT TIME   direct path read             36b84f5s2yj4a   0   36081

1 row  selected.

SQL> /

SID       STATE               EVENT                         SQL_ID         CH#  SEQ#
--------  ------------------- ----------------------------- ------------- ----- -------
152       WAITED SHORT TIME   direct path read             74kfrv5xqpbxf   0   52652

1 row  selected.

SQL> /

SID       STATE               EVENT                         SQL_ID         CH#  SEQ#
--------  ------------------- ----------------------------- ------------- ----- -------
152       WAITED SHORT TIME   direct path read             74kfrv5xqpbxf   0   56786
```

Vamos agora voltar para a sessão 1, e mudar o parâmetro oculto "_serial_direct_read" para NEVER, e executar a mesma consulta, observe agora que não vamos mais ter o evento direct path read, mais sim o db file scattered read, ou seja a nossa consulta estará alocando todo os dados para a SGA:

```
--> SESSÃO 1

SQL>  ALTER SESSION SET "_serial_direct_read"=NEVER;

Session  altered.

SQL>  select avg(length(col1) + length(col2)) from fss.hsk1 where col3 > 1;
 --> SESSÃO 2

SQL>  SELECT
      1  s.sid, w.state, w.event, s.sql_id,  s.sql_child_number, w.seq#
      2  FROM v$session s, v$session_wait w
      3  WHERE w.sid = s.sid AND w.sid=152;

SID       STATE               EVENT                      SQL_ID         CH#  SEQ#
--------  ------------------- -------------------------- ------------- ----- -------
152       WAITED SHORT TIME   db file scattered read    74kfrv5xqpbxf   0   23902

1 row  selected.

SQL> /

SID       STATE               EVENT                      SQL_ID         CH#  SEQ#
--------  ------------------- -------------------------- ------------- ----- -------
152       WAITED SHORT TIME   db file scattered read    74kfrv5xqpbxf   0   26483

1 row  selected.

SQL> /

SID       STATE               EVENT                      SQL_ID         CH#  SEQ#
--------  ------------------- -------------------------- ------------- ----- -------
152       WAITED SHORT TIME   db file scattered read    74kfrv5xqpbxf   0   26977
```

Identificando o uso do Direct Read, através das views de estatísticas

Podemos também identificar o evento direct path através das views v$sesstat e v$mystat, no caso a v$sesstat representa todas as estatísticas de todas as sessões do banco, já a v$mystat representa apenas as estatísticas da minha atual sessão. Diferente da view v$session_wait que mostra o estado atual da sessão, as views de estatísticas são acumulativas para todas as estatísticas. Nesse caso, a estatística chamada "table scans (direct read)" representa a quantidade que o evento direct path foi utilizado dentre todas as instrução realizadas para a mesma sessão.

Pelo motivo das views de estatísticas v$sesstat e v$mystat serem acumulativas, precisamos realizar o antes e o depois e termos um delta para a comparação se aquela sessão sofreu ou não um aumento das estatísticas.

Podemos realizar o teste da seguinte maneira:

```
SQL>  col value format 999999999999999

SQL>  SELECT s.name, m.value
      2  FROM  v$mystat m, v$statname s
      3   WHERE m.statistic# = s.statistic#
      4  AND  s.name = 'table scans (direct read)';

NAME                                             VALUE
-----------------------------------------------  ----------------
table  scans (direct read)                       0

1 row  selected.
```

Veja acima, que a minha sessão está com a estatística "table scans (direct read)" com o valor zerado. Isso mostra que a sessão até o momento não realizou nenhuma leitura de bloco através do evento Direct Path Reads.

Vamos agora, alterar o parâmetro oculto "_SERIAL_DIRECT_READ" para NEVER, afim de forçar a leitura via FULL TABLE SCANS na tabela fss.hsk1 sem a utilização do DPR.

```
SQL>  ALTER SESSION SET "_serial_direct_read"=NEVER;
Session  altered.

SQL>  select avg(length(col1)) from fss.hsk1 where col3 > 1;

AVG(LENGTH(COL1))
-----------------
175.015666

1 row  selected.
```

Após realizar a consulta, vamos novamente realizar a consulta para verificar o valor da estatística “table scans (direct read)”.

```
SQL>  SELECT s.name, m.value
   2  FROM  v$mystat m, v$statname s
   3  WHERE m.statistic# = s.statistic#
   4  AND  s.name = 'table scans (direct read)';

NAME                                              VALUE
------------------------------------------------  ----------------
table  scans (direct read)                        0

1 row  selected.
```

Veja que a estatística continua com o valor zero. Vamos executar a mesma instrução SQL, porém agora forçando o uso do Direct Reads.

```
SQL>  ALTER SESSION SET "_serial_direct_read"=ALWAYS;
Session  altered.

SQL>  select avg(length(col1)) from fss.hsk1 where col3 > 1;

AVG(LENGTH(COL1))
-----------------
175.015666

1 row  selected.
```

Com a execução acima, voltemos a verificar o valor da estatística “table scans (direct read)”.

```
SQL> SELECT s.name, m.value
     2  FROM  v$mystat m, v$statname s
     3   WHERE m.statistic# = s.statistic#
     4  AND  s.name = 'table scans (direct read)';

NAME                                               VALUE
-------------------------------------------------  ----------------
table  scans (direct read)                         1
```

Como você pode ver, a estatística passo de 0 para 1, isso aconteceu porque a instrução foi executada via Direct Path Reads. Para cada consulta então que realizo o evento de Direct Reads, o valor de 1 é adicionado a estatística “table scans (direct read)”. O mesmo procedimento é válido também para a estatística “index fast full scans (direct read)”

```
SQL>  select avg(length(col1)) from fss.hsk1 where col3 > 1;

AVG(LENGTH(COL1))
-----------------
175.015666

1 row  selected.
```

Veja novamente que vamos ter a estatística “table scans (direct read)” com o valor agora de 2.

```
SQL>  SELECT s.name, m.value
      2  FROM  v$mystat m, v$statname s
      3   WHERE m.statistic# = s.statistic#
      4  AND  s.name = 'table scans (direct read)';

NAME                                              VALUE
------------------------------------------------  ----------------
table  scans (direct read)                        2
```

Identificando o uso do Direct Read, através do evento de 10046

Um método muito rápido de identificar também se sua consulta está utilizando “db file scattered read” ao invés de “direct path reads” é através do evento 10046.

Setando esse evento para a sessão, e observando o trace, podemos identificar facilmente se a consulta está sendo feita via “db file scattered read” ou “direct path reads”.

Veja o exemplo abaixo:

```
SQL>  ALTER SESSION SET EVENTS '10046 trace name context forever, level 12'
Session  altered.
```

Com a sessão alterada para o evento 10046, vamos identificar o arquivo de trace da sessão:

```
SQL>  SELECT tracefile
      2    FROM v$process WHERE addr = (
      3    SELECT paddr FROM v$session
      4    WHERE sid = (SELECT sid FROM v$mystat  WHERE rownum < 2)
      5    );

TRACEFILE
------------------------------------------------------------------------------
/u01/app/oracle/diag/rdbms/dbtst/dbtst/trace/dbtst_ora_60173.trc
```

Com os mesmos testes realizados acima, onde forçamos a utilização do Direct Reads, seu arquivo de trace irá se parecer como a listagem abaixo para a instrução com o _serial_direct_read para ALWAYS.

```
WAIT  #140675437128128: nam='direct path read' ela= 780 file number=6 first dba=42624  block cnt=128 obj#=76837
tim=1397656466688788
WAIT  #140675437128128: nam='direct path read' ela= 824 file number=6 first dba=42752  block cnt=128 obj#=76837
tim=1397656466692249
WAIT  #140675437128128: nam='direct path read' ela= 831 file number=6 first dba=42880  block cnt=128 obj#=76837
tim=1397656466696735
WAIT  #140675437128128: nam='direct path read' ela= 757 file number=6 first dba=43008  block cnt=128 obj#=76837
tim=1397656466701094
WAIT  #140675437128128: nam='direct path read' ela= 765 file number=6 first dba=43136  block cnt=128 obj#=76837
tim=1397656466705783
WAIT  #140675437128128: nam='direct path read' ela= 574 file number=6 first dba=43268  block cnt=124 obj#=76837
tim=1397656466708691
WAIT  #140675437128128: nam='direct path read' ela= 590 file number=6 first dba=43392  block cnt=128 obj#=76837
```

```
tim=1397656466711190
WAIT  #140675437128128: nam='direct path read' ela= 568 file number=6 first dba=43520  block cnt=128 obj#=76837
tim=1397656466713200
WAIT  #140675437128128: nam='direct path read' ela= 610 file number=6 first dba=43648  block cnt=128 obj#=76837
tim=1397656466715460
WAIT  #140675437128128: nam='direct path read' ela= 562 file number=6 first dba=43776  block cnt=128 obj#=76837
tim=1397656466718398
WAIT  #140675437128128: nam='direct path read' ela= 524 file number=6 first dba=43904  block cnt=128 obj#=76837
tim=1397656466720576

WAIT  #140675437128128: nam='direct path read' ela= 489 file number=6 first dba=44032  block cnt=128 obj#=76837
tim=1397656466723296
WAIT  #140675437128128: nam='direct path read' ela= 792 file number=6 first dba=44160  block cnt=128 obj#=76837
tim=1397656466726823
WAIT  #140675437128128: nam='direct path read' ela= 726 file number=6 first dba=44292  block cnt=124 obj#=76837
tim=1397656466731733
WAIT  #140675437128128: nam='direct path read' ela= 782 file number=6 first dba=44416  block cnt=128 obj#=76837
tim=1397656466736128
WAIT  #140675437128128: nam='direct path read' ela= 786 file number=6 first dba=44544  block cnt=128 obj#=76837
tim=1397656466740659
WAIT  #140675437128128: nam='direct path read' ela= 621 file number=6 first dba=44672  block cnt=128 obj#=76837
tim=1397656466743702
WAIT  #140675437128128: nam='direct path read' ela= 808 file number=6 first dba=44800  block cnt=128 obj#=76837
tim=1397656466747454
WAIT  #140675437128128: nam='direct path read' ela= 568 file number=6 first dba=44928  block cnt=128 obj#=76837
tim=1397656466751477
WAIT  #140675437128128: nam='direct path read' ela= 553 file number=6 first dba=45056  block cnt=128 obj#=76837
tim=1397656466753675
WAIT  #140675437128128:  nam='direct path read' ela= 579 file number=6 first dba=45184 block cnt=128  obj#=76837
tim=1397656466758527
WAIT  #140675437128128: nam='direct path read' ela= 610 file number=6 first dba=45316  block cnt=124 obj#=76837
tim=1397656466761760
WAIT  #140675437128128: nam='direct path read' ela= 768 file number=6 first dba=45440  block cnt=128 obj#=76837
tim=1397656466765429
WAIT  #140675437128128: nam='direct path read' ela= 751 file number=6 first dba=45568  block cnt=128 obj#=76837
tim=1397656466768958
WAIT  #140675437128128: nam='direct path read' ela= 757 file number=6 first dba=45696  block cnt=128 obj#=76837
tim=1397656466772449
FETCH #140675437128128:c=342947,e=1194482,p=45285,cr=45289,cu=0,mis=0,r=1,dep=0,og=1,plh=3450470040,
tim=1397656466776035
STAT  #140675437128128 id=1 cnt=1 pid=0 pos=1 obj=0 op='SORT AGGREGATE (cr=45289  pr=45285 pw=0 time=1194103 us)'
STAT  #140675437128128 id=2 cnt=799999 pid=1 pos=1 obj=76837 op='TABLE ACCESS FULL  HSK1 (cr=45289 pr=45285 pw=0
time=1363830 us cost=12370 size=145598544  card=799992)'
WAIT  #140675437128128: nam='SQL*Net message from client' ela= 1640 driver  id=1413697536 #bytes=1 p3=0 obj#=76837
tim=1397656466782990
FETCH #140675437128128:c=0,e=223,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=0,plh=3450470040,tim=1397656466785440
WAIT  #140675437128128: nam='SQL*Net message to client' ela= 4 driver id=1413697536  #bytes=1 p3=0 obj#=76837
tim=1397656466788033
```

Agora a mesma execução sem a utilização do Direct Reads, deverá se parecer com o seguinte trace:

(Observe a grande quantidade de "db file scattered read")

```
WAIT  #140675437130048: nam='db file scattered read' ela= 774 file#=6 block#=43396  blocks=128 obj#=76837
tim=1397656143526663
WAIT  #140675437130048: nam='db file scattered read' ela= 476 file#=6 block#=43524  blocks=128 obj#=76837
tim=1397656143536192
WAIT  #140675437130048: nam='db file scattered read' ela= 618 file#=6 block#=43652  blocks=128 obj#=76837
tim=1397656143545942
WAIT  #140675437130048: nam='db file scattered read' ela= 569 file#=6 block#=43780  blocks=128 obj#=76837
tim=1397656143555649
WAIT  #140675437130048: nam='db file scattered read' ela= 504 file#=6 block#=43908  blocks=128 obj#=76837
tim=1397656143564865
WAIT  #140675437130048: nam='db file scattered read' ela= 817 file#=6 block#=44036  blocks=128 obj#=76837
tim=1397656143576862
WAIT  #140675437130048: nam='db file scattered read' ela= 500 file#=6 block#=44164  blocks=124 obj#=76837
tim=1397656143589249
WAIT  #140675437130048: nam='db file scattered read' ela= 721 file#=6 block#=44292  blocks=128 obj#=76837
tim=1397656143602144
WAIT  #140675437130048: nam='db file scattered read' ela= 597 file#=6 block#=44420  blocks=128 obj#=76837
tim=1397656143612393
WAIT  #140675437130048: nam='db file scattered read' ela= 710 file#=6 block#=44548  blocks=128 obj#=76837
tim=1397656143622807
WAIT  #140675437130048: nam='db file scattered read' ela= 790 file#=6 block#=44676  blocks=128 obj#=76837
tim=1397656143631916
WAIT  #140675437130048: nam='db file scattered read' ela= 518 file#=6 block#=44804  blocks=128 obj#=76837
tim=1397656143640901
WAIT  #140675437130048: nam='db file scattered read' ela= 450 file#=6 block#=44932  blocks=128 obj#=76837
tim=1397656143649894
WAIT  #140675437130048: nam='db file scattered read' ela= 998 file#=6 block#=45060  blocks=128 obj#=76837
tim=1397656143661462
WAIT  #140675437130048: nam='db file scattered read' ela= 428 file#=6 block#=45188  blocks=124 obj#=76837
tim=1397656143671014
WAIT  #140675437130048: nam='db file scattered read' ela= 537 file#=6 block#=45316  blocks=128 obj#=76837
tim=1397656143679979
WAIT  #140675437130048:  nam='db file scattered read' ela= 809 file#=6 block#=45444 blocks=128  obj#=76837
tim=1397656143705089
WAIT  #140675437130048: nam='db file scattered read' ela= 743 file#=6 block#=45572  blocks=128 obj#=76837
tim=1397656143714724
WAIT  #140675437130048: nam='db file scattered read' ela= 742 file#=6 block#=45700  blocks=124 obj#=76837
tim=1397656143752173
FETCH #140675437130048:c=13221989,e=13812881,p=45286,cr=45298,cu=0,mis=0,r=1,dep=0,og=1,plh=3450470040,
tim=1397656143762546
STAT  #140675437130048 id=1 cnt=1 pid=0 pos=1 obj=0 op='SORT AGGREGATE (cr=45298  pr=45286 pw=0 time=13812802 us)'
STAT  #140675437130048 id=2 cnt=799999 pid=1 pos=1 obj=76837 op='TABLE ACCESS FULL  HSK1 (cr=45298 pr=45286 pw=0
time=7547357 us cost=12370 size=145598544  card=799992)'
WAIT  #140675437130048: nam='SQL*Net message from client' ela= 1056 driver  id=1413697536 #bytes=1 p3=0 obj#=76837
tim=1397656143767562
FETCH #140675437130048:c=0,e=566,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=0,plh=3450470040,tim=1397656143768778
WAIT  #140675437130048: nam='SQL*Net message to client' ela= 4 driver id=1413697536  #bytes=1 p3=0 obj#=76837
tim=1397656143770042
```

Identificando o uso do Direct Read, via Oracle Internals (apenas para Oracle Geek guys :)

O evento Direct Path Reads é realizado através da função do sistema operacional chamada **kcbldrget**, que significa Kernel Block Direct Read Get.

Para identificar o uso do Direct Path Reads, basta identificar se a função kcbldrget foi disparada do processo do sistema operacional. Isso é possível através do comando pstack do Linux. Com esse comando, podemos identificar stack trace (pila do trace) da execução do processo, ou seja, todo os caminhos via chamadas de SO que o processo passou.

Através do SPID, podemos executar o seguinte procedimento após a execução do mesmo SQL com o parâmetro oculto "_serial_direct_read" para ALWAYS, forçando assim a execução via DPR. No momento da execução, conectado no sistema operacional (no meu caso o Linux), realizamos o comando pstack apontando para o SPID da SESSÃO B, que no caso é o número 50834.

```
[root@oralnx001  ~]# pstack 50834
#0  0x0000003f1960ee33 in __pread_nocancel ()  from /lib64/libpthread.so.0
#1  0x00000000093521fb in skgfqio ()
#2  0x0000000009222e03 in ksfd_skgfqio ()
#3  0x0000000009222b68 in ksfdgo ()
#4  0x000000000234e30e in ksfdaio ()
#5  0x00000000021ef424 in kcflbi ()
#6  0x0000000000ebc90a in kcbldio ()
#7  0x0000000000ebba84 in kcblrs ()
#8  0x0000000000ebb165 in kcblgt ()
#9  0x0000000000eb9941 in kcbldrget ()
#10 0x000000000907b554 in kcbgtcr ()
#11 0x000000000905ff29 in ktrget3 ()
#12 0x000000000905f784 in ktrget2 ()
#13 0x0000000009016ead in kdst_fetch ()
#14 0x0000000000c87f89 in kdstf00000010000kmP ()
#15 0x0000000008ffc6e8 in kdsttgr ()
#16 0x0000000009245970 in qertbFetch ()
#17 0x000000000926cc1f in qergsFetch ()
#18 0x0000000009136e83 in opifch2 ()
#19 0x0000000009140 4e8 in opiefn0 ()
#20 0x000000000914dfc4 in opipls ()
#21 0x000000000913d4d4 in opiodr ()
#22 0x00000000091e7043 in rpidrus ()
#23 0x0000000009354764 in skgmstack ()
#24 0x00000000091e8b5e in rpiswu2 ()
#25 0x00000000091e8188 in rpidrv ()
#26 0x00000000091d14d1 in psddr0 ()
#27 0x00000000091d10e7 in psdnal ()
#28 0x0000000003736b52 in pevm_EXIM ()
#29 0x000000000372831b in pfrinstr_EXIM ()
#30 0x00000000093eae35 in pfrrun_no_tool ()
#31 0x00000000093e9509 in pfrrun ()
#32 0x00000000093f0b61 in plsql_run ()
#33 0x000000000371cb6b in peicnt ()
#34 0x0000000002fa18b1 in kkxexe ()
#35 0x0000000009145 0f9 in opiexe ()
#36 0x0000000001b5cb07 in kpoal8 ()
#37 0x000000000913d4d4 in opiodr ()
#38 0x00000000092e02d6 in ttcpip ()
#39 0x00000000017ece01 in opitsk ()
#40 0x00000000017f19fa in opiino ()
#41 0x000000000913d4d4 in opiodr ()
#42 0x00000000017e8d3c in opidrv ()
#43 0x0000000001de40cb in sou2o ()
#44 0x0000000000a0b0c1 in opimai_real ()
#45 0x0000000001dea03c in ssthrdmain ()
#46 0x0000000000a0b02d in main ()
```

Esse monte de código acima, representa cada chamada de sistema operacional feita pelo processo em execução do SPID 50834. Como você pode ver na linha em negrito, temos uma chamada para o sistema operacional para a função kcbldrget, que significa que a instrução executada por esse processo utilizou o método Direct Reads :)

Condições para se utilizar o Direct Path Reads

Como mencionei acima, a partir do Oracle 11g é muito mais provável que sua consulta utilize o direct path reads, porém existe maneiras de "tentar" identificar se sua consulta irá ou não, utilizar automaticamente o evento.

Além dos processos parallel slaves que são sempre executados via direct path reads, para instruções não paralelas elas funcionam em certas condições. O calculo é baseado em diversos fatores como a quantidade de blocos do objeto e o tamanho do buffer. Existe ainda o parâmetro oculto **_SMALL_TABLE_THRESHOLD** que determina a quantidade mínima de blocos que uma tabela deve ter para a utilização do DPR. O valor default desse parâmetro é 2680, o que significa que caso uma tabela tenha a quantidade de blocos maior que 2680, o seu full table scan será mais favorável a utilizar direct path reads.

Veja um exemplo:

```
SQL>  show parameter _small_table_threshold

NAME                                 VALUE
------------------------------ -------
_small_table_threshold         2680
```

Acima como se pode ver, o parâmetro oculto "_SMALL_TABLE_THRESHOLD" está definido para o valor default de 2680 blocos. Vou agora, criar uma tabela buscando todos os dados da view dba_objects.

```
SQL>  create table t as select * from dba_objects;
Table  created.

SQL>  select blocks from dba_segments where segment_name='T';

BLOCKS
----------
1152
```

Observe que a nossa tabela ficou com 1152 blocos, bem abaixo do valor do parâmetro _small_table_threshold. Vamos executar um FULL TABLE SCAN de encontro a tabela e verificar se foi realizado automaticamente o direct path reads de encontro a tabela.

```
SQL>  select s.name, m.value
      2    from v$statname s, v$mystat m
      3    where m.statistic#=s.statistic#
      4    and s.name = 'table scans (direct read)';

NAME                                               VALUE
-------------------------------------------------  ----------
table  scans (direct read)                         0
```

```
1 rows  selected.

SQL>  select count(*) from t;

COUNT(*)
----------
75214

1 row  selected.

SQL>  select
      2    s.name, m.value
      3    from v$statname s, v$mystat m
      4    where m.statistic#=s.statistic#
      5    and s.name = 'table scans (direct read)';

NAME                                              VALUE
------------------------------------------------  ----------
table  scans (direct read)                        0

1 rows  selected.
```

Veja acima, que a estatística table scans (direct read) não foi alterado. Vamos aumentar a quantidade de blocos da tabela e realizar o mesmo teste.

```
SQL>  insert into t (select * from t);
75214  rows created.

SQL>  insert into t (select * from t);
150428  rows created.

SQL>  commit;
Commit  complete.

SQL>  select blocks from dba_segments where segment_name='T';

BLOCKS
----------
4352

1 row  selected.
```

Agora sim temos blocos acima da quantidade necessária especificada pelo parâmetro _small_table_threshold. Antes de fazer o teste, vamos limpar a área de memória do Oracle para que o teste anterior não interfira nesse novo.

```
SQL>  ALTER SYSTEM FLUSH BUFFER_CACHE;
System  altered.

SQL>  ALTER SYSTEM FLUSH SHARED_POOL;
System  altered.
```

Feito a limpeza, vamos checar novamente a mesma consulta:

```
SQL>  select count(*) from t;

COUNT(*)
----------
300856

1 row  selected.

SQL>  select
      2      s.name, m.value
      3      from v$statname s, v$mystat m
      4      where m.statistic#=s.statistic#
      5      and s.name = 'table scans (direct read)';

NAME                                              VALUE
------------------------------------------------  ----------
table  scans (direct read)                        1

1 row  selected.
```

Agora sim a nossa consulta foi realizada via direct read.

Um outra ponto muito importante para o direct reads, é que ele apenas funciona quando um full scan acontece, ou seja, a função "direct path reads" (kcbldrget) somente é chamada após um full scans. Note que o termo full scans representa os termos TABLE ACCESS FULL e INDEX FAST FULL SCAN no seu plano de execução. Dessa forma, pelo simples fato de uma operação em sua consulta a uma tabela for do tipo UNIQUE SCAN, o Direct Path Reads irá acontecer.

Processos Parallel e o Direct Path Reads

Os processos paralelos são outro ponto importante, como comentei acima, não importa que tipo de processo parallel você tem utilizado (AUTO DOP, IN MEMORY PARALELL, QUEUEING PARALLEL, etc ..) ou o tipo de Degree, a sua consulta sempre que utilizar a operação FULL SCANS ela irá ser feita via Direct Path Reads.

Veja no exemplo abaixo, que mesmo definindo o parâmetro "_serial_direct_read" para "never" vamos ter nossa consulta paralela utilizando o direct reads:

```
SQL>  ALTER SESSION SET "_serial_direct_read"=never;
Session  altered.

SQL>  select
      2    s.name, m.value
      3    from v$statname s, v$mystat m
      4    where m.statistic#=s.statistic#
      5    and s.name = 'table scans (direct read)';

NAME                                              VALUE
------------------------------------------------  ----------
table  scans (direct read)                        0

1 row  selected.

SQL>  select count(*) from t;
```

```
COUNT(*)
----------
300856

1 row  selected.

SQL>  select
      2    s.name, m.value
      3    from v$statname s, v$mystat m
      4    where m.statistic#=s.statistic#
      5    and s.name = 'table scans (direct read)';

NAME                                             VALUE
-----------------------------------------------  --------
table  scans (direct read)                       0

1 row  selected.

SQL>  alter table t parallel 2;
Table  altered.

SQL>  select count(*) from t;

COUNT(*)
----------
300856

1 row  selected.

SQL>  select
      2    s.name, m.value
      3    from v$statname s, v$mystat m
      4    where m.statistic#=s.statistic#
      5    and s.name = 'table scans (direct read)';

NAME                                             VALUE
-----------------------------------------------  --------
table  scans (direct read)                       26

1 row  selected.
```

Ou veja, processos parallel ignoram completamente o parâmetro _serial_direct_read.

Um pouco sobre o parâmetro _serial_direct_read

Você já deve ter entendido bem a utilização do parâmetro oculto _serial_direct_read. Ele força ou ignora a utilização do "Direct Path Reads". Vale a pena lembrar que por se tratar de um parâmetro oculto, não é nem um pouco legal você definir em seu ambiente de produção sem antes consultar a Oracle.

O parâmetro _serial_direct_read ele possuiu esses valores a partir do Oracle 11g:

1 ALWAYS
2 AUTO
3 NEVER
4 TRUE
5 FALSE

Para cada um deles ele trata de um modo diferente a instrução via "Direct Path Reads". O modo default dele é AUTO, o que significa que ele será automático, seguindo toda aquela regra que já expliquei sobre a quantidade e blocos e vários outros fatores.

Através do suporte da Oracle do documento "Exadata Smartscan Is Not Being Used On Insert As Select (Doc ID 1348116.1)", a Oracle descreve um problema em que o Smartscan não acontece devida a não execução da instrução via "Direct Path Reads" (um dos caso mais comum de falta de performance do Exadata). Na nota ele informa o uso do parâmetro _serial_direct_read para TRUE como a resolução do problema.

Joel Perez é um DBA Especialista (Oracle ACE Director, OCM Cloud Admin. & OCM11g). Com mais de 14 anos de experiência do mundo Oracle Technology, especializado em arquitetura e implementação de soluções como: Cloud, Alta disponibilidade, Disaster/Recovery, Upgrades, replicação e todos as áreas relacionadas com bancos de dados Oracle. Consultor internacional com deveres, conferências e atividades em mais de 50 países e inúmeros clientes em todo o mundo. Palestrante regular nos eventos Oracle em todo o mundo como: OTN LAD, OTN MENA, OTN APAC e muito mais. Joel sempre foi conhecido por ser pioneiro em tecnologia Oracle desde os primeiros dias de sua carreira sendo o primeiro latino-americano premiado como "OTN Expert" no ano de 2003 pela Oracle Corporation, um dos primeiros "ACE Oracle" no Oracle ACE Program no ano de 2004, um dos primeiros OCP Database Cloud Administrator em todo o mundo no ano de 2013 e como um das maiores realizações profissionais em sua carreira, recentemente ele foi homenageado como um dos primeiros "OCM Database Cloud Administrator" do mundo.

Flávio Soares é um Oracle DBA Sênior, Exadata DMA, Troubleshooter e Consultor Oracle, certificado em OCP/OCE RAC. Especialista em Exadata, alta disponibilidade e replicação de dados com soluções Oracle. Flávio disponibiliza frequentes informações para a comunidade Oracle através do seu blog.

Este artigo foi revisto pela equipe de produtos Oracle e está em conformidade com as normas e práticas para o uso de produtos Oracle.